UCRL-ID-133248

Simulation of Beamline Alignment Operations

M. G. Miller
C. Annese

February 2, 1999

Simulation of Beamline Alignment Operations

FY98 LDRD Project
Final Summary Report

Mark G. Miller
Cynthia Annese

February 2, 1999

1.0 Introduction

The CORBA-based Simulator was a Laboratory Directed Research and Development (LDRD) project that applied simulation techniques to explore critical questions about distributed control systems. The simulator project used a three-prong approach that studied object-oriented distribution tools, computer network modeling, and simulation of key control system scenarios. The National Ignition Facility's (NIF) optical alignment system was modeled to study control system operations.

The alignment of NIF's 192 beamlines is a large complex operation involving more than 100 computer systems and 8000 mechanized devices. The alignment process is defined by a detailed set of procedures; however, many of the steps are deterministic. The alignment steps for a poorly aligned component are similar to that of a nearly aligned component; however, additional operations/iterations are required to complete the process. Thus, the same alignment operations will require variable amounts of time to perform depending on the current alignment condition as well as other factors. Simulation of the alignment process is necessary to understand beamline alignment time requirements and how shared resources such as the Output Sensor and Target Alignment Sensor effect alignment efficiency.

The simulation has provided alignment time estimates and other results based on documented alignment procedures and alignment experience gained in the laboratory. Computer communication time, mechanical hardware actuation times, image processing algorithm execution times, etc. have been experimentally determined and incorporated into the model.

Previous analysis of alignment operations utilized average implementation times for all alignment operations. Resource sharing becomes rather simple to model when only average values are used. The time required to actually implement the many individual alignment operations will be quite dynamic. The simulation model estimates the time to complete an operation using distributions rather than static values. The only way to accurately understand resource utilization and time requirements for a complex industrial application such as alignment, is to utilize simulation tools such as Simprocess to model the system.

1.1 Overview of alignment systems

The alignment system is comprised of several types of computer systems and many different types of mechanical devices as illustrated in Figure 1. The Automatic Alignment Front-End Processor (AAFEP) controls the alignment sequence by commanding other systems to perform specific functions. In a simplistic view, the AAFEP first requests the Video FEP (VFEP) to acquire video images and then analyzes the images to determine the current alignment state of a particular device. After calculating the alignment error, the AAFEP requests the Alignment Controls FEP (ACFEP) to manipulate hardware devices in precisely defined methods to correct the misalignment.

All communications between these computer systems is through the Integrated Computer Control System (ICCS) Network, which is comprised of 100BaseT Ethernet using the CORBA protocol, and ATM. The single AAFEP is centrally located on the network, while the approximately 100 ACFEPs and 28 VFEPs are distributed throughout the NIF bays to remain close to their hardware devices.

Error! No topic specified.

Figure 1 Overview of the alignment system and its many computer systems

1.2 Simulation goals

The primary simulation goal is to validate the one-hour alignment time requirement specified in the Automatic Alignment Subsystem Design Requirements (SSDR). The alignment process is a combination of many sequential and concurrent operations. The use of shared resources, dynamic alignment processes, and dynamic implementation times makes it difficult to estimate the total alignment time without using a simulation tool.

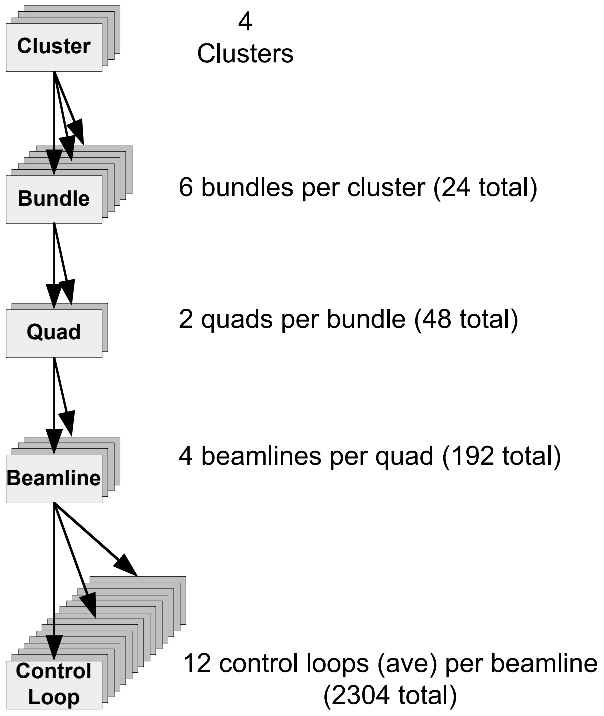
A secondary goal is to understand resource utilization and how they effect the alignment time. Most of the FEPs and shared resources mentioned above are utilized at low average levels during the alignment process. At times, many are utilized at very high levels and will become unavailable for various amounts of time. These loading effects must be analyzed to ensure that the available resources are adequately sized and configured.

2.0 Simulation Model

2.1 Top View

NIF is comprised of 192 beamlines that must be aligned prior to each laser shot. The structuring of the system is broken into several hierarchical layers shown in Figure 2. Each cluster, bundle, and quad is aligned concurrently. The four beamlines that form a quad are also concurrently aligned; however, they share large amounts of hardware and thus have significant interactions. Each quad contains 18 unique control loops that must be sequentially implemented within each beamline.

A quad is a relatively independent section of the system and was subsequently modeled in this simulation. To produce alignment results for the entire system, the interaction between quads must also be incorporated in the model and the simulation must be run multiple times since only 1/48 of the system is modeled.



Cluster
4 Clusters
Bundle
6 bundles per cluster (24 total)
Quad
2 quads per bundle (48 total)
Beamline
4 beamlines per quad (192 total)
Control Loop
12 control loops (ave) per beamline (2304 total)

Figure 2 Hierarchical structure of the NIF laser system

2.2 Component Alignment

The alignment of a component consists of sequentially applying one or more control loops to the component as illustrated in Figure 3. If a single control loop is required, the other two control loops are simply bypassed. For a centering-pointing control loop, the centering and pointing control loops must be repeated several times. The actual number of iteration is dependent upon the current alignment condition of the component and linearity of the cross-coupling matrix. The triangular distribution TRI (0.8, 2.0, 3.2) is a typical example, where the range is 0.8 to 3.2 and the average is 2.0 iterations.

Error! No topic specified.

Figure 3 Illustrates the control loops in each component

2.3 Control Loop

A control loop block diagram is shown in Figure 4. The large number of different control loops has necessitated using a single template to create all types. Individual initialization files are used to create unique behavior/operations for each one. Each portion of the control loop is explained in detail in the following sections.

Error! No topic specified.

Figure 4 Simulation model for a control loop.

2.3.1 Reference Setup and Alignment Setup

During each control loop, a particular light source must be visible by the appropriate camera. A set of devices, such as shutters, pinholes, lenses, and mirrors must be positioned correctly. An initialization file for each control loop type specifies the time to complete the manipulation of the all devices and the time to complete the multiple CORBA calls.

2.3.2 Acquire Image

The image acquisition distribution models the time required for the AAFEP to generate the CORBA call to the Video FEP, wait for the next video sync pulse, acquire the image from the CCD camera, and return the video image to the AAFEP via a CORBA return call. Each type of CCD camera can be assigned a unique acquisition time in the simulation. A single triangular distribution, TRI (0.18, 0.2,0.22), was used for each of the cameras initially; however, individual values can be incorporated in the future.

2.3.3 Process Image

Image processing algorithms are typically applied twice during a control loop - once to analyze the reference image and again to analyze the alignment state image. The simulation model specifies several different algorithms to be implemented in the various control loops.

Studies have been performed to determine which type of algorithm is required for most control loops. In general, a weighted centroid will be calculated from each image after it has been filtered/smoothed. This process requires the application of two FFTs and several simple algorithms. The algorithms have been optimized and benchmarked on a computer system identical to the AAFEP. Execution times of each algorithm are utilized in the model.

2.3.4 Adjust Devices

The component's mirrors are adjusted multiple times during each control loop. Laboratory testing and manufacturer specifications were used to determine the range of adjustment times. Each type of device has been assigned a distribution that has average implementation times that range from 2 to 7 seconds.

2.3.5 Control Loop Iterations

A control loop must be repeated multiple times until the alignment error is within a specified tolerance. Laboratory tests on a scaled model of NIF have provided good data on control loop behavior. The number of iterations used in the model is based on several distributions that have average values that range from 4 to 7 iterations.

2.3.6 Corba Call

A series of tests have been performed to measure the time required to generate and return CORBA messages. Type of computer systems and size of message were both considered during these tests. The following table outlines the values used in this simulation.

Table 1 Implementation times Corba messages

FEP Communication	Message Size	Time to complete CORBA call and return
AAFEP- ACFEP	256 bytes	1.7-2.3 msec
AAFEP-VFEP	310Kbytes	180-220 msec

2.4 Interactions between quads

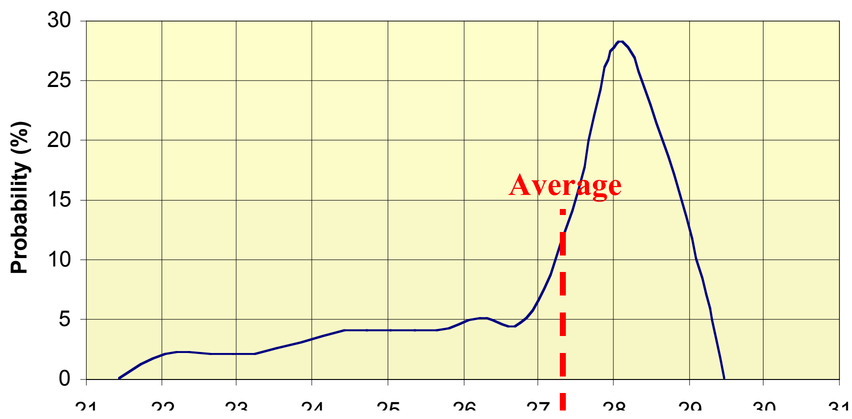
A single set of sensors collectively known as the Target Alignment Sensor (TAS) is used to point all beamlines onto the target. This shared resource is reserved by beamlines when they are ready to acquire an image. Due to the great number of beamlines that require its use, the sensor is largely unavailable. The entire target alignment process has been incorporated into the model to simulate TAS utilization of the remaining 47 Quads.

3.0 Results

3.1 Alignment operations

The simulation of the entire alignment system, which includes the PAM, PABTS, Main Laser, Switchyard, Target Area and both the TAS and target positioners, requires 29.3 minutes to complete. It assumes beamlines are well behaved and operator interaction is not required. Since the overall alignment time is equal to the time required to align the slowest beamline, and the slowest beamline has a fairly high probability (10% for each of the 48 Quads) of requiring 29.0 to 29.3 minutes to complete, the simulation results are quite consistent.

The alignment time for individual quads, shown in Figure 5, varies from 21.5 to 29.3 minutes with an average of 27.2 minutes. As stated above, alignment is completed when the last quad has finished the alignment process. When Quads are able to complete alignment early, the TAS shared resource becomes increasingly available to the remaining beamlines, and thus they are also aligned more quickly. When a Quad requires additional time to complete the early stages of alignment, it can “catch up” since it has

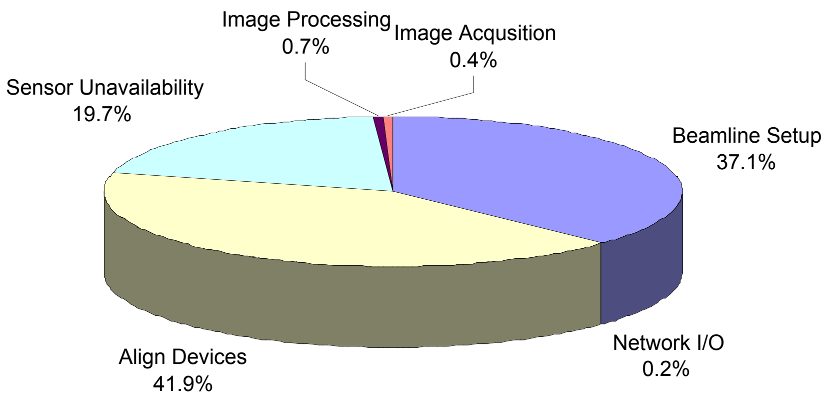


sole utilization of the resource and therefor seldom requires significant additional time to complete. For these reasons, the quad alignment times vary widely for those below the average time, but the alignment times are closely bunched for those above the average.

The alignment time from the simulation results is less than those predicted from the static alignment model. The dynamic simulation model was able to more efficiently utilize shared resources. Because there are time variations when a beamline requires a shared resource (primarily the Target Alignment Sensor which is shared among all 192 beamlines), the utilization of these sensors was started earlier than previously estimated and thus was subsequently more available to the beamlines over a longer period of time. The shared resources were thus increasingly available to the Quads that required longer alignment operations. The single largest fear was that the utilization of shared resources would be less efficient than the static model suggested and would require additional time to implement. It is interesting to note that this fear was unfounded and ultimately led to reduced alignment time.

3.2 Resource utilization

The complex alignment operations can be broken into several key tasks which are common to each type of control loop: beampath setup, image acquisition, image analysis, alignment corrections, and network communications. The chart shown in Figure 6 illustrates the time required to implement these tasks. The alignment time is dominated by the actuation times of mechanical devices required to both setup the beampath and adjust the alignment hardware. Significant time was also spent waiting for the TAS to become available.



This data assumes the alignment of all beamlines. Since many of the alignment operations are performed concurrently, especially when the beamlines are pointing or focusing on to the target, this data does not represent critical path operations. If the time required to adjust the alignment devices was reduced to half the value, the overall alignment time would not be reduced by 21% (50% of the 42% dedicated to device adjustment).

Although image processing is only a small portion of the overall alignment time, it dominates the CPU utilization of the Automatic Alignment FEP. When the simulation results of the Quad alignment are scaled to model the entire NIF, image processing

Figure 6 Alignment time is dominated by actuation times of mechanical devices

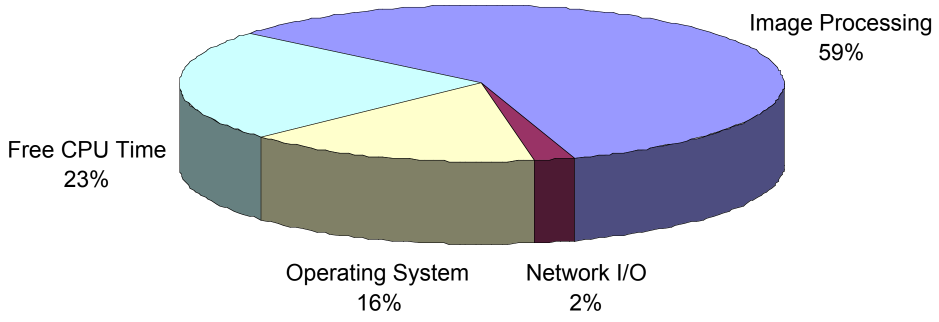


Figure 7 Processor time utilization shows CPU resources do not limit operations

requires approximately 59% of the CPU time. Through computer benchmarks and knowledge of operating systems overhead, CPU utilization was estimated and is illustrated in Figure 7.

4.0 Difficulties

The original scope of the alignment simulation was significantly greater than the model implemented. It was intended to model the entire 192 beamlines and assign resources to many additional devices. After a significant portion was modeled, it became apparent that the model was too large for Simprocess to run. Large models can be built and saved, however, there seems to be a file size limitation (8Mbytes) for successful simulation. This limitation forced the model to incorporate only a single quad or 1/48 of the system and use less resource allocations. Although the overall alignment results are just as valid, the level of detail was reduced.

The complete alignment model could have provided better resource modeling, especially the TAS. Operational details of the Alignment Controls and Video FEPs would have been useful information to validate their requirements. Individual device utilization such as mirrors, reticles, pinholes, shutters would have also been useful to identify cycle times and number of operations each device would endure during their thirty-year life cycle.

ModSim may have been better simulation tool to use for this application. The large number of expressions required to build the ideal model would require a heartier but less elegant simulation process. ModSim is another simulation tool produced by CACI that relies on scripts rather than GUI tools. It is much like writing a program in the C language. Complex routines can be created and reused much like a procedure or function. Since only a small portion of ModSim was included in the Simprocess package, it would not support stand-alone simulations.

* Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48